ET420077155US

APPLICATION

FOR

UNITED STATES LETTERS PATENT

INTERNATIONAL BUSINESS MACHINES CORPORATION

A METHOD OF TESTING A COMPUTER PROGRAM TRANSLATED INTO A NATIONAL LANGUAGE

Field of Invention

The present invention relates to a method of testing a computer program translated into a national language.

Background of the Invention

A computer program generally interacts with a user showing several user-visible text messages, such as menus, help panels, and the like. Most programs are originally developed in the English language. When the program must be distributed to countries in which the English is not the national language, it is a common practice to translate all text messages into the national language of the country.

The translated program must be tested in order to ensure that the text messages are correctly translated for the context in which they appear; moreover, it is necessary to ensure that the appearance of the translated text messages is correct, for example that the translated text messages within lists and push buttons have an appropriate size. The test (known as Translation Verification Test, or TVT) is commonly carried out by translators who are native of the country in which the program must be distributed.

The testers usually do not have technical expertise. As a consequence, the testers are required to move to a software development laboratory, wherein hardware,

software and product experts are available; these technical experts help the testers to set up a testing environment, and to install, configure and drive the translated program in order to show to the testers all the translated text messages.

This implies a large waste of time for moving the testers to the development laboratory; moreover, the moving of each tester involves high costs, such as for the travelling and the stay. All of the above greatly limits the use of the test process and prevents it from being used extensively, with a consequent reduction in the level of quality and reliability of the software production.

It is an object of the present invention to overcome the above mentioned drawbacks. In order to achieve this object, a method of testing a computer program translated into a national language as set out in the first claim is proposed.

Disclosure of the Invention

Briefly, the present invention provides a method of testing a computer program translated into a national language comprising the steps of making a package in a first location, the package including the translated program, an operating environment for the translated program, a sample of input data for the translated program, a set of translated scripts defining test commands for showing user-visible text messages of the

translated program, and a play module for automatically executing the test commands, sending the package to a tester at a second location remote from the first location, operating a testing computer at the second location with the package, and running the translated program on the testing computer using the test commands.

Moreover, the present invention also provides a corresponding system for testing a computer program translated into a national language, a package for use in this system, and a computer readable medium of a removable type storing the package.

Brief Description of the Drawings

Further features and the advantages of the solution according to the present invention will be made clear by the following description of a preferred embodiment thereof, given purely by way of a non-restrictive indication, with reference to the attached figures, in which:

Fig.1 is a basic block diagram of a system in which the method of the invention can be used,

Fig.2 shows a structure under test,

Fig.3 depicts the content of a hard-disk in a development computer of the system,

Fig.4a-4b are a flow chart of a method used for testing the translated program.

Description of the Preferred Embodiments

With reference in particular to Fig. 1, there is shown a system 100 comprising a computer 105d, for example a PC (Personal Computer), which is installed at a software development laboratory 110d. The development computer 105d has several units, which are connected in parallel to a communication bus 115d. In particular, a central processing unit (CPU) 120d controls operation of the development computer 105d, a working memory 125d (typically a DRAM) is used directly by the CPU 120d, and a read-only memory (ROM) 130d stores a basic program for starting the development computer 105d. Various peripheral units are further connected to the bus 115d (by means of respective interfaces). Particularly, a bulk memory consists of a hard-disk 135d and of a driver unit (DRV) 140d for reading/writing CD-ROMs; the development computer 105d further includes an input unit (IN) 145d, which consists for example of a keyboard and a mouse, an output unit (OUT) 150d, which consists for example of a monitor and a printer, and a network interface card (NIC) 155d.

A CD-ROM 160 is inserted into the driver unit 140d and it is burnt; the CD-ROM 160 is then sent by an air-mail service 165 to a further computer 105t installed at a testing location 110t, which is remote from the development laboratory 105d (typically in a different country). The testing computer 105t is similar to the development computer 110d. Particularly, the testing computer 105t includes a communication bus 115t, a central processing unit (CPU) 120t, a working memory

(DRAM) 125t, a read-only memory (ROM) 130t, a hard-disk 135t, a driver unit (DRV) 140t, an input unit (IN) 145t, an output unit (OUT) 150t, and a network interface card (NIC) 155t. The computers 105d and 105t consist of units of the same type (for example keyboards with the same national configuration), even if some minor differences are possible (such as in the capacity of the hard-disks).

Similar considerations apply if the computers include different units (for example a driver unit for floppy-disks or a scanner), if the computers have a different structure (such as with a multi-processor architecture), if the development computer and the testing computer are different (but preferably of a compatible type, so that any software can run substantially in the same manner on both computers), if the CD-ROM is sent to the testing location by express courier (or any other delivery service), and the like.

Computer programs are developed by software engineers at the laboratory 110d. The programs are generally written in an object-oriented language and they have a Graphical User Interface (GUI), which allows a user to control the program by using stylised screen objects, such as windows, dialog boxes, pop-up or pull-down menus, text panels, and push or radio buttons. The user interacts with the program by generating a sequence of mouse and keyboard events, to which the program must respond.

As shown in Fig.2, a program originally developed at the laboratory consists of a functional module (EXEC) 205 and a configuration module (EN_CONF) 210e. The functional module 205 embodies the definition (data structures and methods) of each class of objects used by the original program, and the procedures for generating instances of said objects and calling the respective methods in response to events generated by the user. The configuration module 210e contains a set of dialog descriptors, each one consisting of a file that describes dialog (window or dialog box) contents. Particularly, the dialog descriptor includes dialog attributes (for example, a dialog's title as shown in a dialog's title bar) and a list of GUI-objects presented by the dialog and the relevant attributes (for example, a pushbutton and a pushbutton caption as visible in the Graphical User Interface). All user-visible text messages associated with each dialog and with each contained GUI-object are written in an original language, typically English.

A set of test scripts is used to verify the appearance of the text messages of the original program 205,210e. The test scripts consist of a command module (CMD) 215 and a declaration module (EN_DCL) 220e. The command module 215 embodies a test case defined by a sequence of commands (written in a scripting language by a script writer), which simulate user actions, such as pushing of a button or typing a key; each GUI-object is referenced in the test commands by a symbolic identifier. The declaration module 220e consists of a file containing

records that map the symbolic identifiers with the respective GUI-object's actual identifiers. For windows and dialog boxes, the actual identifier is the title (caption) as it appears to the user, whereas the contained GUI-objects (such as, push buttons and text fields) are identified by an index (i.e., the order of the GUI-object in relation to its sibling objects). Ultimately, the declaration module 220e contains data structures similar to the ones that are contained in the configuration module 210e. Each window and dialog box, which is described in the declaration module 220e, contains a field pointing to the file name containing the relevant dialog descriptor (configuration module 210e).

The original program 205,210e is translated into one or more languages different from English, in order to localise the program for use in foreign countries. Each translated program, also known as a National Language Support (NLS) version of the program, consists of the same functional module 205 of the original program, which is associated with a configuration module (NLS_CONF) 210t; the configuration module 210t embodies all the dialog descriptors in the corresponding national language (such as Italian).

The (translated) configuration module 210t and the (original) declaration module 220e are provided to a translation module (TRS) 235, which generates a translated declaration module 220t; the translated declaration module 220t is similar to the original

declaration module 220e, with the (dialog) actual identifiers translated into Italian. Particularly, the translation module 235 parses the original declaration module 220e. Each time the translation module 235 identifies a statement declaring a window or a dialog box, the same data structure is created into the translated declaration module 220t; the translated actual identifier (caption) is extracted from the translated configuration module 210t, and it is inserted into the corresponding field of the translated declaration module 220t.

For example, a dialog box named "File Open" and containing a push button named "Exit" is defined in the (original) configuration module 210e as follows:

```
Command Dialog {
Attributes {
Title = "File Open"; }
Gadgets {
     CommandButton {
          Title = "Exit"; }
}
}
```

The original declaration module 220e for the same GUI-object contains the following statements:

```
Window DialogBox FO
tag "File Open"
const
```

PushButton EX

Tag "#1"

The command in the module 215, which simulates the user action of clicking the mouse on the "Exit" button, is:

```
FO.EX.Click()
```

wherein the symbolic identifiers FO is associated with the actual identifier "File Open" and the symbolic identifier EX is associated with the first PushButton within the dialog box.

The translated configuration module 210t of the Italian version of the program is obtained from the original configuration module 210e by replacing the (English) names "File Open","Exit" with the corresponding (Italian) names "Apri File","Esci":

```
Command Dialog {
Attributes {
Title = "Apri File"; }
Gadgets {
     CommandButton {
          Title = "Esci"; }
}
}
```

The translated declaration module 220t is then obtained from the original declaration module 220e by replacing the name "File Open" with the corresponding

name "Apri File", which is extracted from the translated configuration module 210t.

Likewise considerations apply if the programs and the test scripts have a different structure, if the program is written in a conventional language and has a Character User Interface (CUI), if the (original) test scripts are obtained by directly recording a sequence of events generated by the user, if the original program and the translated program are written in different national languages, if a unique configuration module serves both the functional module and the command module (in this way, the translation of the program under test has no impact on the test scripts), and so on.

Considering now Fig.3, the hard-disk 135d (of the development computer at the laboratory) stores an operating system (NLS_OS) 305 in the version corresponding to the language of the translated program 205,210t (Italian in the example at issue). The operating system 305 consists of several modules, for example controlling basic functions of the development computer, a network access and an e-mail service, which as a whole define an operating environment for the translated program 205,210t.

The translated program 205,210t is then installed onto the hard-disk 135d, together with the corresponding (translated) test scripts 215,220t. The translated program 205,210t is populated with a sample of input data

(IN_DT) 310. A play-back module (PB) 315 is input the command module 215 and the translated declaration module 220t. The play-back module 315 resolves the symbolic identifiers contained in the command module 215, replacing them with the corresponding translated actual identifiers (read from the translated declaration module 220t); the play-back module 315 drives the translated program 205,210t transforming the resolved test commands into GUI-event streams that cause the requested actions to occur. The hard disk also includes a test driver module (TST) 317, which invokes the translated test scripts (to be executed by the play-back module 315) and prompts the tester to verify the relevant text messages, which are subject to the test process.

The hard-disk 135d further includes a back-up module 320. The back-up module 320 takes a snap-shot of the hard-disk 135d and generates a package 325, which consists of a file containing the operating system 305, the translated program 205,210t, the translated test scripts 215,220t, the input data 310, the play-back module 315, and the test driver module 317 in a compressed form. The package 325 and a module 330 for restoring the package 325 are provided to a driver module (DRV) 335, which physically controls the burning of the CD-ROM.

Similar considerations apply if the programs and the data are organised in a different manner, if other

functions are provided, if the package has a different structure, and the like.

With reference now to Figg.4a-4b, when a new program must be shipped, a series of routines, which together make up a method 400, are performed at successive stages in time. The method starts at block 405 and then goes to block 410, wherein the original program is developed at the laboratory (with the text messages in the original configuration module written in English). Proceeding to block 415, the original test scripts (for verifying the translation and appearance of the text messages) are generated. The translated program is obtained from the original program at block 420 by translating all text messages (in the configuration module) into Italian. The method continues to block 425, wherein the translated (dialog) actual identifiers are extracted from the translated configuration module, and then passes to block 430, wherein the translated actual identifiers are inserted into the respective fields of the translated declaration module. Descending into block 435, the operating system, the translated program, the translated test scripts, the input data, the play-back module, the test driver module, and the back-up module are installed onto the hard-disk of the development computer. The back-up module takes a snap-shot of the hard-disk at block 440, and generates the package containing the operating system, the translated program, the translated test scripts, the input data, the play-back module, and the test driver module in a compressed form. Going now to

block 442, the package and the restoring module are recorded onto a CD-ROM. The CD-ROM is sent to the testing location by the air-mail service at block 445.

Considering now block 450, the CD-ROM is received at the testing location (in Italy in the example at issue) by a tester, which speaks Italian as his mother tongue. The tester inserts the CD-ROM in the driver unit of the testing computer at block 455, and switches on the same; the tester computer is then booted from the CD-ROM, which involves loading the operating system and starting the restoring module stored thereon. As soon as the tester confirms the execution of the restoring module, the method proceeds to block 457, wherein the package is de-compressed and installed onto the hard-disk of the testing computer (completely wiping out any previous data). The testing computer is switched off at block 460; the tester removes the CD-ROM and then restarts the testing computer.

Passing to block 465, the bootstrap of the testing computer (from the hard-disk) involves the automatic execution of the test driver module, which drives the translated program using the translated test scripts executed by the play-back module. The play-back module simulates user actions, which cause a text message to be show to the tester at block 470. The tester is prompted at block 475 to verify whether the meaning of the text message and its appearance are correct. If so, the method continues to block 480 (described in the following).

Conversely, if a defect is detected by the tester, the method descends into block 485, wherein a corresponding entry (containing the detected defect along with a name of the test script that revealed the error) is written to a problem reporting file stored on the hard-disk of the testing computer; the method then continues to block 480. In both cases, the method checks at block 480 if the end of the test scripts has been reached. If not, the method returns to block 470. On the contrary, the method descends into block 490, wherein an e-mail with the problem reporting file attached thereto is sent to the development laboratory. The method then ends at the final block 495.

Likewise considerations apply if an equivalent method is carried out, for example with routines for controlling the execution of the test process, for selecting the test commands to be run, for storing a point reached in the test process as a starting point for a next running of the test scripts, for selecting the test scripts to be run or run again a previously executed test script, and the like.

More generally, in the method of testing a computer program translated into a national language according to the present invention, a package including the translated program, an operating environment for the translated program, a sample of input data for the translated program, a set of translated scripts defining test commands for showing user-visible text messages of the

translated program, and a play module for automatically executing the test commands is made in a first location. The package is sent to a tester at a second location remote from the first location; a testing computer at the second location is operated with the package, and the translated program is then run on the testing computer using the test commands.

The solution of the invention removes the need for testers to move to the development laboratory; this allows the costs for the travelling and the stay of the testers to be saved.

Therefore, the test process is more efficient and cost effective; the test process can be used on a large scale, thereby increasing the level of quality and reliability in the software production.

The preferred embodiment of the present invention described above offers further advantages. Particularly, the package obtained taking a snap-shot of the bulk memory of the development computer onto the CD-ROM guarantees that an exact disk image of the development computer, with the translated program already installed, preconfigurated and populated with significant data, is used on the testing computer. Moreover, the testing computer is booted from the CD-ROM and the package is then restored onto its hard-disk. Therefore, the same testing environment envisaged at the development laboratory is used by the tester; this result is obtained

in a simple manner and without requiring any technical expertise on the part of the tester.

The tester is automatically prompted to verify each text message; the detected defects are recorded onto the problem reporting file, which is then sent to the development laboratory by e-mail. These features ensure that the tester verifies all text messages; moreover, they make it very easy for the tester to report the results of the test to the development laboratory.

Likewise considerations apply if the package is stored onto a floppy-disk (or any other computer readable medium of a removable type), if it is restored onto the hard-disk of the testing computer with a different procedure, if the problem reporting file is sent to the development laboratory in a different manner (for example by a floppy-disk); alternatively, the package is created in a different manner, it is sent to the testing computer by e-mail, it is not restored onto the testing computer, or the tester is not prompted to verify each text message.

The translated test scripts are automatically obtained from the corresponding original test scripts; preferably, the process is carried out replacing the original (dialog) actual identifiers with the corresponding translated actual identifiers in the declaration module. This allows the program to be tested in different languages from a single set of test scripts

(with the declaration module in English). Therefore, the creation, maintenance and portability of the test scripts are greatly enhanced.

Similar considerations apply if the programs and the test scripts have a different structure, if the translated test scripts are obtained from the corresponding original test scripts in a different manner, and so on. However, the solution of the present invention leads itself to be implemented even directly writing or recording the test scripts for each translated program.

Naturally, in order to satisfy local and specific requirements, a person skilled in the art may apply to the solution described above many modifications and alterations all of which, however, are included within the scope of protection of the invention as defined by the following claims.